

AMENDMENTS TO THE CLAIMS

1. (Currently Amended) A method in a computer system having a global descriptor table register for executing code during a system management mode interrupt (SMI), the method comprising:
upon occurrence of an SMI~~system management mode interrupt~~,
saving state of the computer system;
switching the computer system to protected mode;
replacing first contents of the global descriptor table register that point to a first global descriptor table in use when the system management mode interrupt occurred with second contents that point to a second global descriptor table that is distinct from the first global descriptor table;
executing 32-bit code that uses a using the second global descriptor table that is different from the global descriptor table in use when the system management mode interrupt occurred; and
upon completion of the execution of the 32-bit code,
restoring the saved state of the computer system; and
returning from the occurrence of the SMI~~interrupt~~.
2. (Previously Presented) The method of claim 1 wherein the 32-bit code is an operating system kernel for loading and running programs during the occurrence of the system management mode interrupt.
3. (Previously Presented) The method of claim 2 wherein the programs are Windows Portable Executable programs.
4. (Previously Presented) The method of claim 1 wherein the computer system is based on an Intel Pentium processor.

5. (Currently Amended) A method in a computer system having a global descriptor table register for executing code during an SMI~~system management mode interrupt~~, the method comprising:

upon occurrence of an SMI~~system management mode interrupt~~,

switching the computer system from the system management mode to another~~a second~~ mode;

replacing ~~first~~ contents of the global descriptor table register that point to a first global descriptor table in use when the system management mode interrupt occurred with ~~second~~ contents that point to a second global descriptor table that is distinct from the first global descriptor table; and

~~executing code that uses a~~ using the ~~second~~ global descriptor table ~~that is different from the~~ global descriptor table in use when the system management mode interrupt occurred.

6. (Currently Amended) The method of claim 5 including upon completion of the execution of the code, returning from the occurrence of the SMI~~system management mode interrupt~~.

7. (Currently Amended) The method of claim 5 including:

saving state of the computer system; and

upon completion of the execution of the code,

restoring the saved state of the computer system; and

returning from the occurrence of the SMI~~system management mode interrupt~~.

8. (Previously Presented) The method of claim 5 wherein the executing executes 32-bit code.

9. (Currently Amended) The method of claim 5 wherein the code is an operating system kernel for loading and running programs during the occurrence of the SMIsystem management mode interrupt.

10. (Previously Presented) The method of claim 9 wherein the programs are portable executables.

11. (Previously Presented) The method of claim 5 wherein the computer system is based on an Intel processor.

12. (Previously Presented) The method of claim 5 wherein the computer system is based on an Intel-compatible processor.

13. (Currently Amended) The method of claim 5 wherein the programs are executed code is selected from the group consisting of a remote console program, a remote boot program, a remote diagnostics program, a remote restart program, and a debugging program.

14. (Currently Amended) The method of claim 5 wherein the ether second mode is protected mode.

15. (Currently Amended) The method of claim 5 wherein the computer system has a foreground operating system, and wherein the code executes transparently to the foreground operating system.

16. (Currently Amended) The method of claim 5 wherein the computer system has a foreground operating system, and wherein the code executes even if the foreground operating system has crashed or stopped.

17. (Currently Amended) The method of claim 5 wherein the computer system has a foreground operating system, and wherein the code executes when the foreground operating system crashes or stops.

18. (Currently Amended) A computer-readable medium containing instructions for an SMI system management mode interrupt routine that allows execution of code that resides in physical address space above address 0x100000above a 1MB boundary, by a method comprising:

saving state of the processor;

switching the processor to protected mode; and

before returning from the SMI interrupt, executing code that resides in physical address space above address 0x100000is stored above the 1MB boundary.

19. (Currently Amended) The computer-readable medium of claim 18 including after executing the code, restoring the saved state of the processor and returning from the SMI interrupt.

20. (Currently Amended) The computer-readable medium of claim 18 wherein the code is executed using a global descriptor table that is different from the global descriptor table in use when the SMI interrupt occurred.

21. (Previously Presented) The computer-readable medium of claim 18 wherein the processor is a Pentium-based processor.

22. (Previously Presented) The computer-readable medium of claim 18 wherein the executed code is 32-bit flat address space code.

23. (Previously Presented) The computer-readable medium of claim 18 wherein the instructions are loaded into system management memory by a BIOS.

24. (Previously Presented) The computer-readable medium of claim 18 wherein the code is loaded into memory from a ROM.

25. (Currently Amended) The computer-readable medium of claim 18 wherein the code is loaded into memory from a ~~f~~Flash ROM.

26. (New) The method of claim 1 wherein a processor switches to system management mode and executes an SMI in response to a signal received on a package pin of the processor.

27. (New) The method of claim 26 wherein the package pin is the SMI# package pin.

28. (New) The method of claim 1 wherein a processor switches to system management mode and executes an SMI in response to a message received via an APIC bus of the processor.

29. (New) The method for claim 1 wherein a processor switches to system management mode and executes an SMI in response to a message received via a front side bus of the processor.

30. (New) The method of claim 1 wherein a processor chip set is the source of the SMI.

31. (New) The method of claim 1 wherein a Northbridge controller is the source of the SMI.

32. (New) The method of claim 1 wherein a Southbridge controller is the source of the SMI.

33. (New) The method of claim 1 wherein an electronic circuit is the source of the SMI.

34. (New) The method of claim 1 wherein returning from the occurrence of the interrupt is accomplished by executing an RSM instruction.

35. (New) The method of claim 1, further comprising, upon completion of the 32-bit code, changing the contents of a hardware register indicating a cause that triggered the SMI.

36. (New) The method of claim 5 wherein a processor switches to system management mode and executes an SMI in response to a signal received on a package pin of the processor.

37. (New) The method of claim 36 wherein the package pin is the SMI# package pin.

38. (New) The method of claim 5 wherein a processor switches to system management mode and executes an SMI in response to a message received via an APIC bus of the processor.

39. (New) The method of claim 5 wherein a processor switches to system management mode and executes an SMI in response to a message received via a front side bus of the processor.

40. (New) The method of claim 5 wherein a processor chip set is the source of the SMI.

41. (New) The method of claim 5 wherein a Northbridge controller is the source of the SMI.

42. (New) The method of claim 5 wherein a Southbridge controller is the source of the SMI.

43. (New) The method of claim 5 wherein an electronic circuit is the source of the SMI.

44. (New) The method of claim 5, further comprising returning from the occurrence of the interrupt by executing an RSM instruction.

45. (New) The method of claim 5, further comprising, upon completion of the executed code, changing the contents of a hardware register indicating a cause that triggered the SMI.

46. (New) The computer-readable medium of claim 18 wherein the method is performed in response to the receipt of an SMI by a processor.

47. (New) The computer-readable medium of claim 46 wherein a processor switches to system management mode and executes an SMI in response to a signal received on a package pin of the processor.

48. (New) The computer-readable medium of claim 47 wherein the package pin is the SMI# package pin.

49. (New) The computer-readable medium of claim 46 wherein a processor switches to system management mode and executes an SMI in response to a message received via an APIC bus of the processor.

50. (New) The computer-readable medium of claim 46 wherein a processor switches to system management mode and executes an SMI in response to a message received via a front side bus of the processor.

51. (New) The computer-readable medium of claim 46 wherein a processor chip set is the source of the SMI.

52. (New) The computer-readable medium of claim 46 wherein a Northbridge controller is the source of the SMI.

53. (New) The computer-readable medium of claim 46 wherein a Southbridge controller is the source of the SMI.

54. (New) The computer-readable medium of claim 46 wherein an electronic circuit is the source of the SMI.

55. (New) The computer-readable medium of claim 18, the method further comprising returning from the interrupt by executing an RSM instruction.

56. (New) The computer-readable medium of claim 18, the method further comprising, upon completion of the executed code, changing the contents of a hardware register indicating a cause that triggered the SMI.